Team SAY
SANCHIT GARG
ABHIRAM SIDDANTHI
YASH HARALE

FACE MASK DETECTION USING OPENCV

22th May 2023

OVERVIEW

Using image processing techniques and using a machine learning model, we have written an algorithm which detects whether subject is wearing a face mask or not. We used OpenCV for implementing face detection and used a machine learning model to draft a conclusion and display it.

PYTHON CODE FOR FACE DETECTION – PART 1

```
import cv2
import os
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA

with_mask = np.load('with_mask.npy')
without_mask = np.load('without_mask.npy')

with_mask = with_mask.reshape(200, 50 * 50 * 3)
without_mask = without_mask.reshape(200, 50 * 50 * 3)

X = np.r_[with_mask, without_mask]
labels = np.zeros(X.shape[0])
labels[200:] = 1.0

pca = PCA(n_components=3)
x_train, x_test, y_train, y_test = train_test_split(X, labels, test_size = 0.25)
x_train = pca.fit_transform(x_train)
svm = SVC()
svm.fit(x_train, y_train)
x_test = pca.transform(x_test)
y_pred = svm.predict(x_test)
```

```
names = {0 : 'Mask', 1 : 'No Mask'}


cascPath = os.path.dirname(
    cv2.__file__) + "/data/haarcascade_frontalface_alt2.xml"
faceCascade = cv2.CascadeClassifier(cascPath)
capture = cv2.VideoCapture(0)

while True:
    flag, img = capture.read()
    faces = faceCascade.detectMultiScale(img)
    for (x,y,w,h) in faces:
        cv2.rectangle(img, (x, y), (x + w, y + h),(255,0,255), 4)
        face = img[y:y+h, x:x+w, :]
        face = cv2.resize(face, (50,50))
        face = face.reshape(1, -1)
        face = pca.transform(face)
        pred = svm.predict(face)[0]
        n = names[int(pred)]
        print(n)
    cv2.imshow('Video', img)
    if cv2.waitKey(1) & 0xFF == 27:
        break



capture.release()
cv2.destroyAllWindows()
```

SPECIFICATIONS FOR 1st PART OF CODE

Importing the necessary libraries import

cv2, os, numpy, SVC, train_test_split, PCA

This line imports the OpenCV library, which is used for image and video processing. The second line imports os which is used in getting the path directory for haar used in face-detection. We import numpy so we can perform mathematical operations on our images. From sklearn library we import SVC which is used in data prediction from the trained model. From sklearn library we also import train_test_split which is used in retrieving the training and testing parameters to find accurate results. We also import the PCA library from sklearn which helps us change the dimension of data to fit our input parameters.

Initializing the machine learning model

with_mask = np.load('with_mask.npy'):

without_mask = np.load('without_mask.npy'):

These load in the training data as numpy files which were previously created using the part two algorithm and we reshape it into 3 dimensions to fit our datatype, We then initialize our training data in X as an array.

Labelling the data

labels = np.zeros(X.shape[0])

labels[200:] = 1.0

Here we label the data we have collected. We are assigning the first 200 images as masked values or 0 and the next 200 as no mask images or 1.0.

Training the data

pca = PCA(n_components=3)

x_train, x_test, y_train, y_test = train_test_split(X, labels, test_size = 0.25)

x_train = pca.fit_transform(x_train)

svm = SVC()

svm.fit(x_train, y_train)

x_test = pca.transform(x_test)

y_pred = svm.predict(x_test)

Here we define a variable called pca and initialize it to have 3 components. Then we retrieve the values of the data x_train, x_test, y_train, y_test from the train_test_split function. We then transform the x_train data to a size that is suitable for the function and we initialize svm to be SVC(). We then fit the training data to x_train to y_train and then use the test data after transforming it and use to to finally predict a y value which we store in y_pred. This is the system we will use to predict whether the subject is wearing a face mask or not.

Capturing of video

```
while True:

  flag, img = capture.read()

  faces = faceCascade.detectMultiScale(img)

  for (x,y,w,h) in faces:

    cv2.rectangle(img, (x, y), (x + w, y + h),(255,0,255), 4)

    face = img[y:y+h, x:x+w, :]
```

The above code captures each frame using capture.read() where capture was defined to be cv2.VideoCapture(0). The frames it is recording are stored in img and using the face detection function: faceCascade.detectMultiScale(img) the faces are detected in the frame and stored in faces. We then take the x,y,w, and h information from the data stored in faces and use it to draw a rectangle around the detected face and we store the face separately in another variable called face so as to perform analysis on the face.

Predicting the result

```
face = cv2.resize(face, (50,50))

face = face.reshape(1, -1)

face = pca.transform(face)

pred = svm.predict(face)[0]

n = names[int(pred)]

print(n)
```

The above code uses the data stored in the face variable and after processing it to be the right size and contain the correct number of dimensions it is then passed into the svm.predict() function which gives out an output and converting this prediction to an integer gives the value of the labels that we had assigned. If the value is 0 then we print that subject is wearing a mask, otherwise if the value is 1 then the subject is not wearing a mask.

PYTHON CODE FOR SAVING TRAINING DATA – PART 2

```
import cv2
import os
import numpy as np

data = []

cascPath = os.path.dirname(
    cv2.__file__) + "/data/haarcascade_frontalface_alt2.xml"
faceCascade = cv2.CascadeClassifier(cascPath)

while True:
    img = cv2.imread('./test/nomask.png')
    faces = faceCascade.detectMultiScale(img)
    for (x,y,w,h) in faces:
        cv2.rectangle(img, (x, y), (x + w, y + h),(255,0,255), 4)
        face = img[y:y+h, x:x+w, :]
        face = cv2.resize(face, (50,50))
        print(len(data))
        if len(data) < 400:
            data.append(face)

    cv2.imshow('Video', img)
    if cv2.waitKey(1) & 0xFF == 27 or len(data) >= 50:
        break

while True:
    img = cv2.imread('./test/guy.jpg')
    faces = faceCascade.detectMultiScale(img)
    for (x,y,w,h) in faces:
        cv2.rectangle(img, (x, y), (x + w, y + h),(255,0,255), 4)
        face = img[y:y+h, x:x+w, :]
        face = cv2.resize(face, (50,50))
        print(len(data))
        if len(data) >= 50:
            data.append(face)

    cv2.imshow('Video', img)
    if cv2.waitKey(1) & 0xFF == 27 or len(data) >= 100:
        break

while True:
    img = cv2.imread('./test/guy3.jpg')
    faces = faceCascade.detectMultiScale(img)
    for (x,y,w,h) in faces:
        cv2.rectangle(img, (x, y), (x + w, y + h),(255,0,255), 4)
```

```
        face = img[y:y+h, x:x+w, :]
        face = cv2.resize(face, (50,50))
        print(len(data))
        if len(data) >= 100:
            data.append(face)

    cv2.imshow('Video', img)
    if cv2.waitKey(1) & 0xFF == 27 or len(data) >= 150:
        break

while True:
    img = cv2.imread('./test/guy1.jpg')
    faces = faceCascade.detectMultiScale(img)
    for (x,y,w,h) in faces:
        cv2.rectangle(img, (x, y), (x + w, y + h),(255,0,255), 4)
        face = img[y:y+h, x:x+w, :]
        face = cv2.resize(face, (50,50))
        print(len(data))
        if len(data) >=150:
            data.append(face)

    cv2.imshow('Video', img)
    if cv2.waitKey(1) & 0xFF == 27 or len(data) >= 200:
        break

np.save('without_mask.npy' ,data)
capture.release()
cv2.destroyAllWindows()
```

SPECIFICATIONS FOR 2ND PART OF CODE

Importing the necessary libraries import

cv2, os and numpy

This line imports the OpenCV library, which is used for image and video processing. We import os so that we can access the directory containing the xml file used in face detection. We also import numpy which can be used to perform complex mathematical calculations on the images.

Initializing data capture

```
Data = []
```

We use this to initialize an empty array to pass in the collected data

Face detection

```
cascPath = os.path.dirname(

    cv2.__file__) + "/data/haarcascade_frontalface_alt2.xml"

faceCascade = cv2.CascadeClassifier(cascPath)
```

We get the location of the required xml file which is used to detect faces and it's path is stored in a variable names cascPath. Then we initialize the variable faceCascade as cv2.CascadeClassifier(cascPath) which we will use to detect faces in the images that we will pass to store as the training data.

Starting loops, capturing data and storage

```
while True:

        face = img[y:y+h, x:x+w, :]

        if len(data) >=150:

                data.append(face)
```

This loop is always running and is only broken when the length of the data array exceeds a certain value. We use 4 such loops in our code but more loops always result in more accurate results. Each loop in here is fed a particular image and the face is recognized and saved into the data array. After the end of 4 loops the data in the data array is saved as a numpy file of subjects not wearing a mask. We can also do the same thing to save the data for subjects wearing a mask.

Data is stored at the end of the loops using the line **np.save('without_mask.npy' ,data**

CHALLENGES FACED

1. Compatibility issues between different libraries and was not able to get TensorFlow to run so had to look for different methods.
2. Issue with identifying faces in lower lighting was difficult and it was difficult to get program to detect faces wearing masks.
3. It was a little difficult trying to understand and reuse the machine learning algorithm
4. It was initially difficult to use more than one image in the training data set but eventually we were able to modify the code to do so.
5. Giving the correct size and dimension to the input data was initially difficult and debugging the errors was a fair challenge